Contribution ID: **363** Type: **not specified**

Folio Restructuring for Huge Pages in guest_memfd

Thursday 11 December 2025 15:45 (15 minutes)

There is active development on adding huge page support to guest_memfd to improve performance of CoCo VMs, specifically around obtaining huge pages from HugeTLB and from the normal buddy allocator in the form of Transparent Huge Pages. Huge page support relies heavily on the ability to restructure pages, to be able to track page users on a per-page basis, using struct page refcounts.

The process of folio restructuring for guest_memfd has many similarities, but also some notable differences, with restructuring of THP pages for other users out of guest_memfd. This session will focus on highlighting overlaps and exploring how to, or whether it makes sense to extend existing folio restructuring for guest_memfd.

Some possible discussions:

- The kernel supports folio_split() for buddy-allocated folios, but merging is only handled on the freeing path. What should merging look like, to support CoCo VMs, without freeing and re-allocating?
- How to extend restructuring to 1G pages? e.g. xas_split_alloc() seems to be limited to certain orders.
- folio_split() handles regular folios; what would extending that for HugeTLB (and retaining HugeTLB Vmemmap Optimization) look like?
  - How would a non-uniform split work with HVO? Can HVO be re-optimized directly from 1G to 2M instead of first requiring 1G to 4K deoptimization and then re-optimizing from 4K to 1G?
- Potential future optmizations:
  - Is there a better way to solve refcounting from all users of a page other than splitting the page?
  - Is there a better way to track users of a page other than refcounts?

Primary author: TNG, Ackerley

Presenter: TNG, Ackerley

Session Classification: Kernel Memory Management MC

Track Classification: Kernel Memory Management MC